REMARKS

This is intended as a full and complete response to the Office Action dated March 27, 2008, having a shortened statutory period for response set to expire on June 27, 2008. Please reconsider the claims pending in the application for reasons discussed below.

Claims 1-22 are pending in the application. Claims 1-13 remain pending following entry of this response. Claims 1 and 6 have been amended. Claims 14-22 have been canceled. Applicants submit that the amendments do not introduce new matter.

Interview Summary

On May 14, 2008, a telephonic interview was held between Gero G. McClellan (attorney of record), Shandon Herring (technical advisor), and the Examiner. The parties discussed the cited reference, *Sparks*. Claims 1 and 6 were discussed. While no agreement as to an allowance was reached, the Examiner acknowledged that a response using the formal drawings to clarify the invention should move prosecution forward. To that end, Applicants submit this response with reference to the formal drawings.

Claim Rejections - 35 U.S.C. § 102

Claims 1-8 and 11 are rejected under 35 U.S.C. 102(b) as being anticipated by *Sparks et al.* (U.S. Patent No. 5,355,469, hereinafter, "*Sparks*").

Applicants respectfully traverse this rejection.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as

is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990).

In this case, *Sparks* does not disclose "each and every element as set forth in the claim." For example, *Sparks* does not disclose the first element of Claim 1. In part, Claim 1 recites, "allowing a user to establish a relationship between one or more of the memory deallocators and one or more of the memory allocators, wherein the relationship requires that memory space allocated by the one or more allocators is freed by the one or more deallocator." Claim 6 recites similar limitations. The Examiner argues that *Sparks* discloses this element at Col. 3, Lines 14-31. However, the cited passage is in fact directed to monitoring memory allocation through the use of a table containing all allocations and deallocations of a program and using that table and other "validity checks" to determine if allocations or deallocations can be made. There is no reference whatsoever to establishing "a relationship between one or more of the memory deallocators and one or more of the memory allocators, wherein the relationship requires that memory space allocated by the one or more allocators is freed by the one or more deallocators."

To illustrate this point, it may be helpful to discuss one embodiment of the current invention using Fig. 4 and Fig. 10. In Fig. 4, an illustrative allocated memory block **400** is shown. The value contained in the tag field **304** is set to one (1) indicating that the memory block **400** is allocated. In addition to the tag field **304**, the allocated memory block **400** also illustrates a size field **402** and an allocation statement field **404**. The field **404** contains information (referred to herein as the "allocation statement") used by the debugger to uniquely determine where in the source code the memory block was allocated. The allocation statement is the allocator of the memory block **400**.

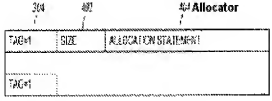

Fig. 4

Paragraph [0056] of the application provides a more in-depth description of a schematic representation of an allocated memory block created using this embodiment of the invention.

Fig. 10 provides a data structure showing an example of a user-established relationship between deallocators and allocators as recited in Claim 1. In Fig. 10, an illustrative producer-consumer table **1000** is shown. The table **1000** comprises an array of pointers to records **1004**. Each entry **1002** of the table **1000** contains a call to a deallocator. A symbolic representation to a deallocator at line **4** of Procedure A is shown in a first entry **1002** of the table **1000**. Each of the records **1004** associated with each deallocator includes a field **1006** containing an allocator statement.

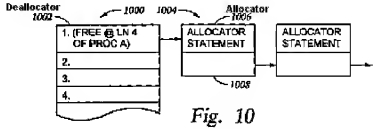

Fig. 10

When an allocation occurs, the heap manager takes steps to write an allocation statement into the field **404** of an allocated memory block **400** (described above with reference to FIG. 4) for each instance of memory allocation. Upon deallocation, a relationship validity check is performed. The validity check compares the allocation statement contained in the field **404** with the allocation statement in the field **1006** of each of the records **1004** according to the particular deallocator called to free the memory block **400**. If the allocation statement contained in the field **404** cannot be matched with one of the allocation statements of the records **1004** the user is notified. For a more in depth description of the embodiment above, read paragraphs [0069]-[0072] of the application.

Sparks simply does not disclose this type of relationship or even this level of memory management. *Sparks* is merely concerned with the **functions** performed by the deallocators and allocators (that is, the deallocations and allocations made by the

deallocators and allocators), rather than the deallocators and allocators, per se. More significantly, *Sparks* make no reference of any kind to establishing relationships between deallocators and allocators. In fact, a simple word search of the cited portions of *Sparks* reveals no instances of the words "relationship", "allocator", or "deallocator". It follows, therefore, that *Sparks* does not disclose, "upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violated," as recited by Claim 1. The Examiner argues that *Sparks* discloses "determining whether the relationship is violated" in the Abstract of Sparks. However, the cited passage from *Sparks* is merely directed to the general idea of automatically detecting errors made by erroneous memory allocations and deallocations with reference to a table structure containing the allocations and deallocations of a program. There is no reference whatsoever to determining whether a relationship between an allocator and a deallocator is violated upon a call to a deallocator. Therefore, *Sparks* does not disclose, "upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violated," as recited by Claim 1. Claim 6 recites similar limitations.

Accordingly, Applicants submit that Claims 1, 6 and their dependents are allowable and respectfully request allowance of the claims.

Additionally, *Sparks* does not disclose "each and every element" as set forth in Claim 8. For example, *Sparks* does not disclose "setting an upper limit on the amount of memory space an allocator can allocate...wherein the upper limit is specific to the allocator," as recited by Claim 8. Instead, *Sparks* merely discloses a user setting the maximum number of concurrent memory allocations in a program which is then used to set the size of the table structure containing memory allocations. Nowhere does *Sparks* mention "setting an upper limit on the amount of memory space an allocator can allocate...wherein the upper limit is specific to the allocator," as recited by Claim 8. Simply stated, *Sparks* does not place any memory space constraints on individual memory allocators. Further, *Sparks* does not monitor or track such constraints on allocators. In fact, a simple word search of *Sparks* reveals no instances of the word

"allocator" at all. Similarly, the other elements of Claim 8, including "tracking the amount of memory space allocated by the allocator;" "and when the amount of memory space allocated exceeds the limit, notifying a user", are not disclosed by *Sparks*. Accordingly, *Sparks* does not disclose each and every limitation of Claim 8. Therefore, Claim 8 and its dependents are believed to be allowable, and allowance of the claims is respectfully requested.

Claim Rejections - 35 U.S.C. § 103

Claims 9-10 and 12-13 are rejected under 35 U.S.C 103(a) as being unpatentable over *Sparks* as applied to Claim 8 above, and in further view of *Sato et al.* (U.S. Patent No. 6,467,075 B1, hereinafter, "*Sato*").

Applicants respectfully traverse this rejection.

Claims 9-10 and 12-13 depend from Claim 8. For the reasons given above with respect to Claim 8, Applicants submit that Claims 9-10 and 12-13 are also necessarily allowable.

Therefore, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

Conclusion

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/

Gero G. McClellan

Registration No. 44,227
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicants